1.) Implement an algorithm to find the kth to last element of a singly linked list. Assume the size of the linked list is not initially known.

The next node in the LinkedList with node n can be accessed using “n.next”.

Give a brief description of your initial approach to this problem? What is its Big O?

```
public LinkedListNode kthToLast(LinkedListNode head, int k) {
```

2.) Complete the `getCopyWithoutTarget` instance method for the `LinkedList314` class. The method returns a copy of the calling object, except any values in the original list equal to a given target value are not included. The relative order of elements not equal to the target is the same between the result and the original list.

**You may not use any other methods in the `LinkedList314` class unless you implement them yourself as a part of your solution.**

The `LinkedList314` class uses singly linked nodes.

- The list has a reference to the first node in the chain of nodes.
- When the list is empty, `first` stores `null`.
- The list does not store `null` data values.
- If the list is not empty, the last node in the chain of nodes, has its next reference set to `null`.
- You may use the nested `Node` class, the `equals` method for `Objects`, and the default `LinkedList314` constructor.
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {
      private Node<E> first;
      private static class Node<E> { // The nested Node class.
            private E data;
            private Node<E> next;
            public Node(E d) {  data = d; }
      }

}
```

Examples of calls to `getCopyWithoutTarget (E tgt)` on various lists. The initial list is on the left and the list returned by the method is on the right In these examples the lists contain `Integer` objects.

```
[], tgt 5 -> returns []
[5], tgt 5 -> returns []
[5, 5, 1, 5, 5], tgt 5 -> returns [1]
[5, 5, 5, 5], tgt 5 -> returns []
[5, 7, 5, 3, 5, 3], tgt 5 -> returns [7, 3, 3]
[7, 7, 3, 12, 15, 0], tgt 5 -> returns [7, 7, 3, 12, 15, 0]
```

```
/* pre: tgt != null
   post: Per the problem description. This list is not altered
as a
     result of this method. */
public LinkedList314<E> getCopyWithoutTarget(E tgt) {
```

3.) Implement an algorithm to delete a node in the middle of a singly linked list, given ONLY access to that node (i.e any node in the list aside from the first or the last). If it cannot be deleted, return false.

public boolean deleteNode(LinkedList n) {